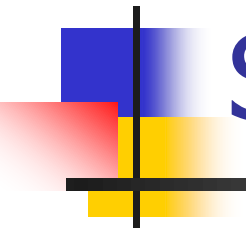
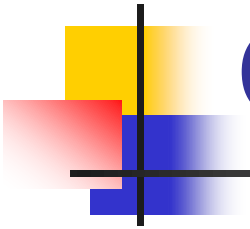


Software Engineering 3

Lecture 5 – Git

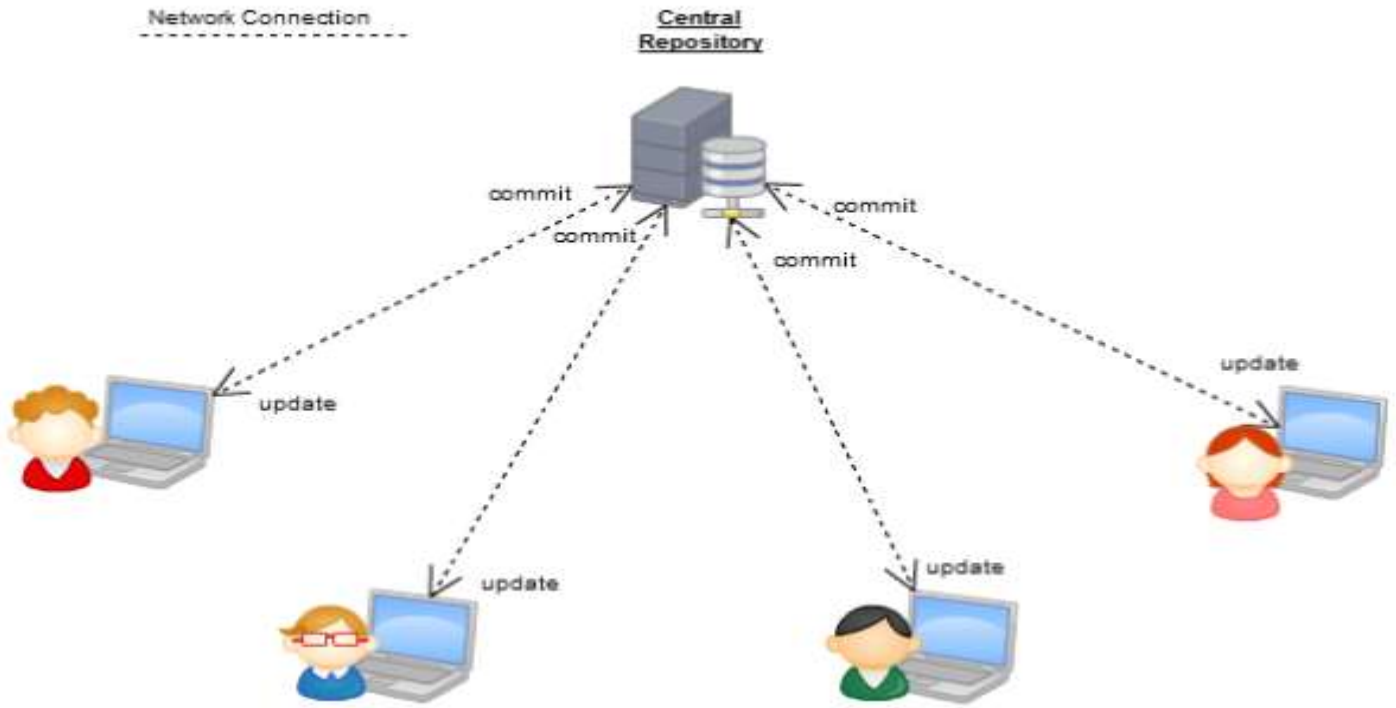
Eng. Sally Jarkas
Eng Aya Joumaa



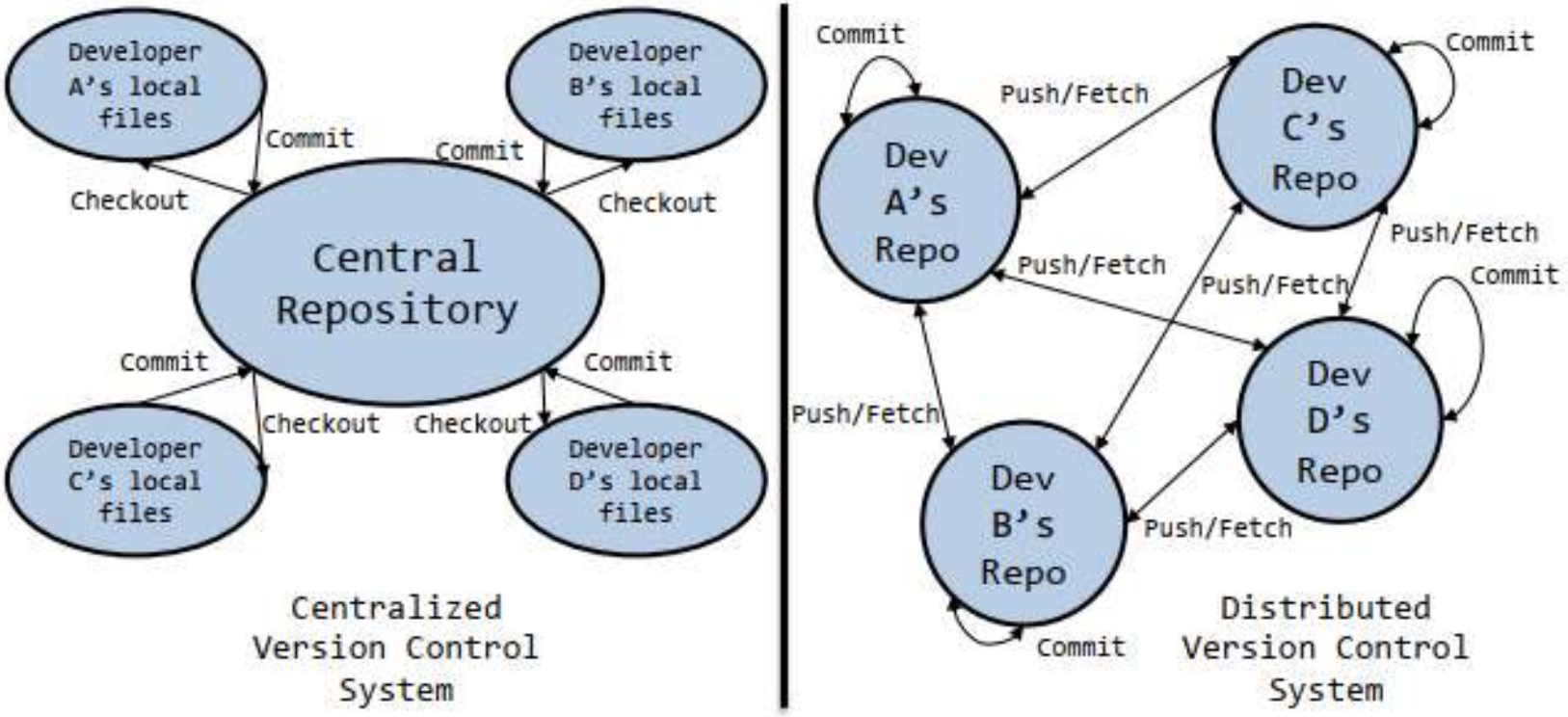
Goals of Version Control

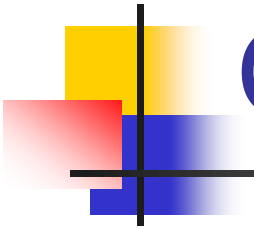
- Be able to search through revision history and retrieve previous versions of any file in a project
- Be able to share changes with collaborators on a project
- Be able to confidently make large changes to existing files

Centralized Version Control Systems



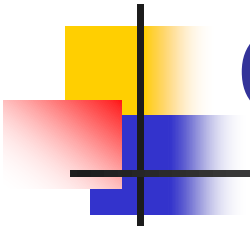
Distributed Version Control Systems (DVCS)





Git

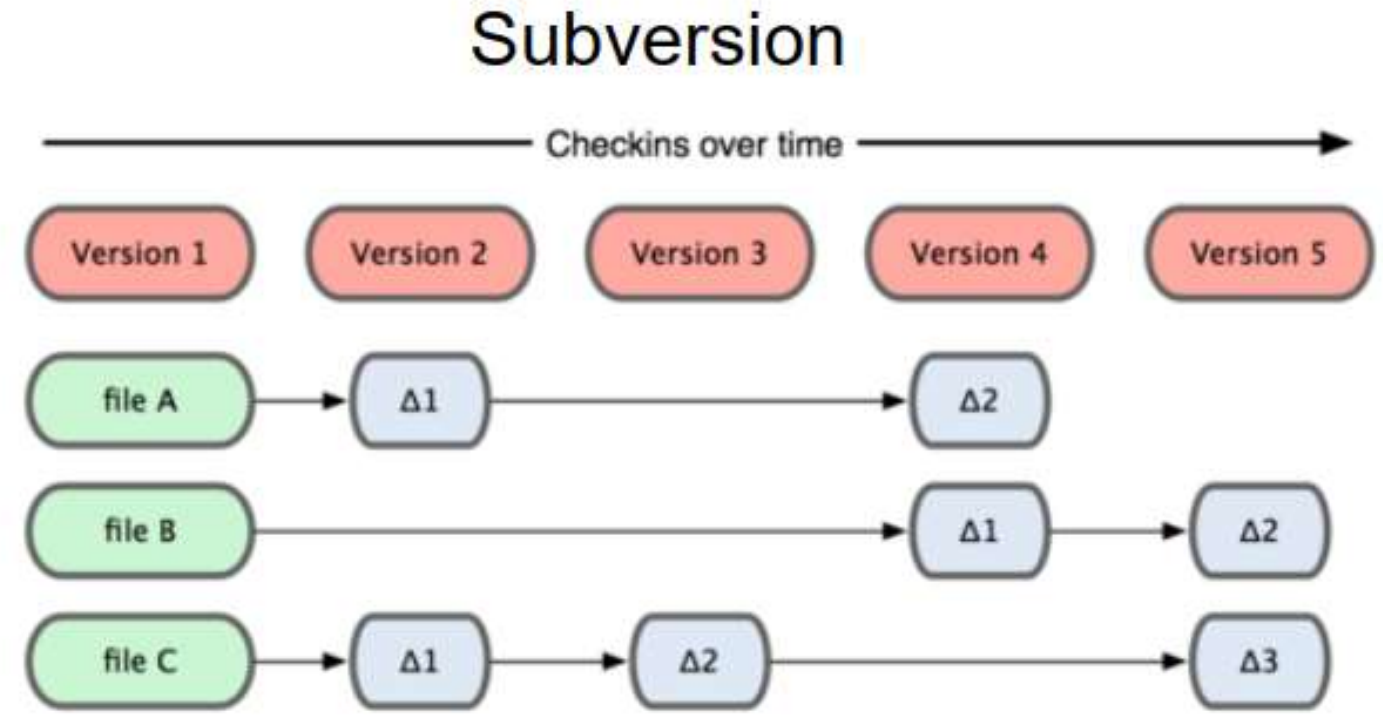
- Created in 2005 by Linus Torvalds to maintain the Linux kernel.
  and he created that too.
- Distributed VCS
  https://www.git-scm.com/



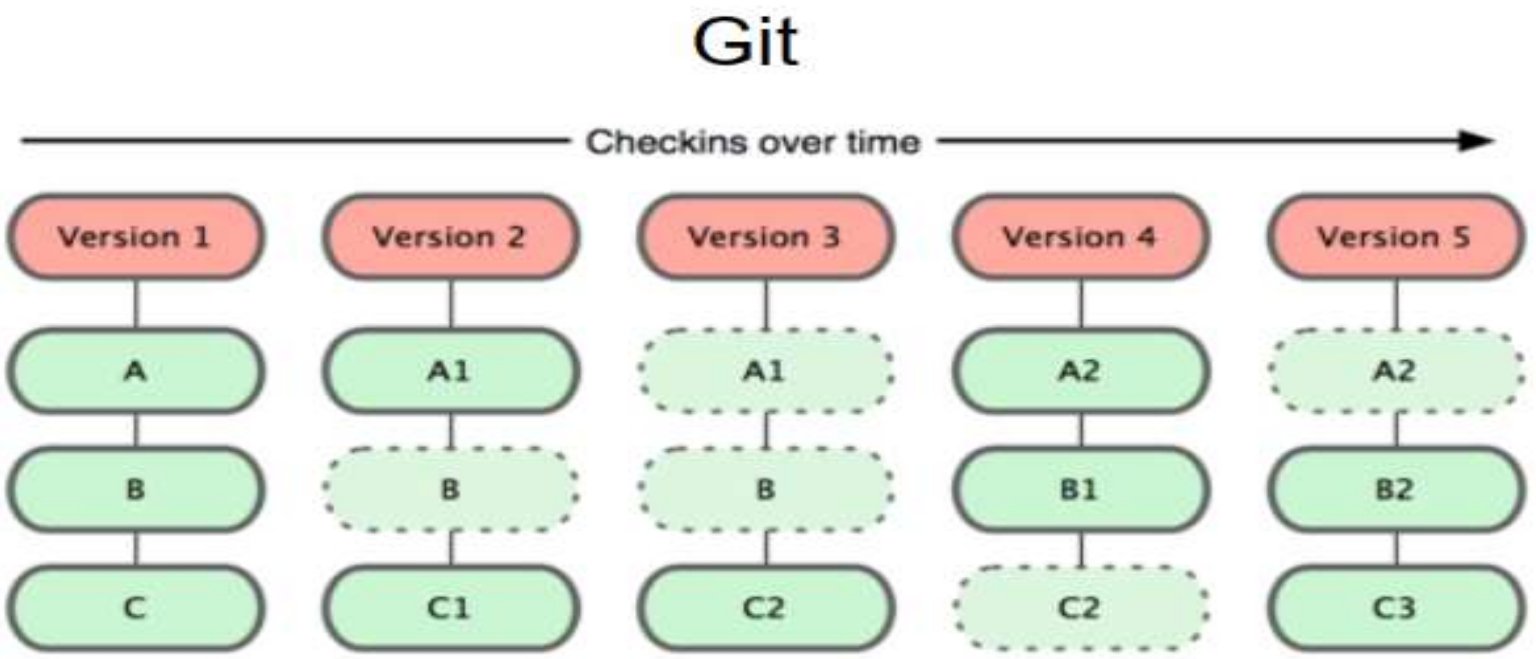
Git History

- Initial goals:
  - Speed
  - Support for non-linear development (thousands of parallel branches)
  - Fully distributed
  - Able to handle large projects like Linux efficiently

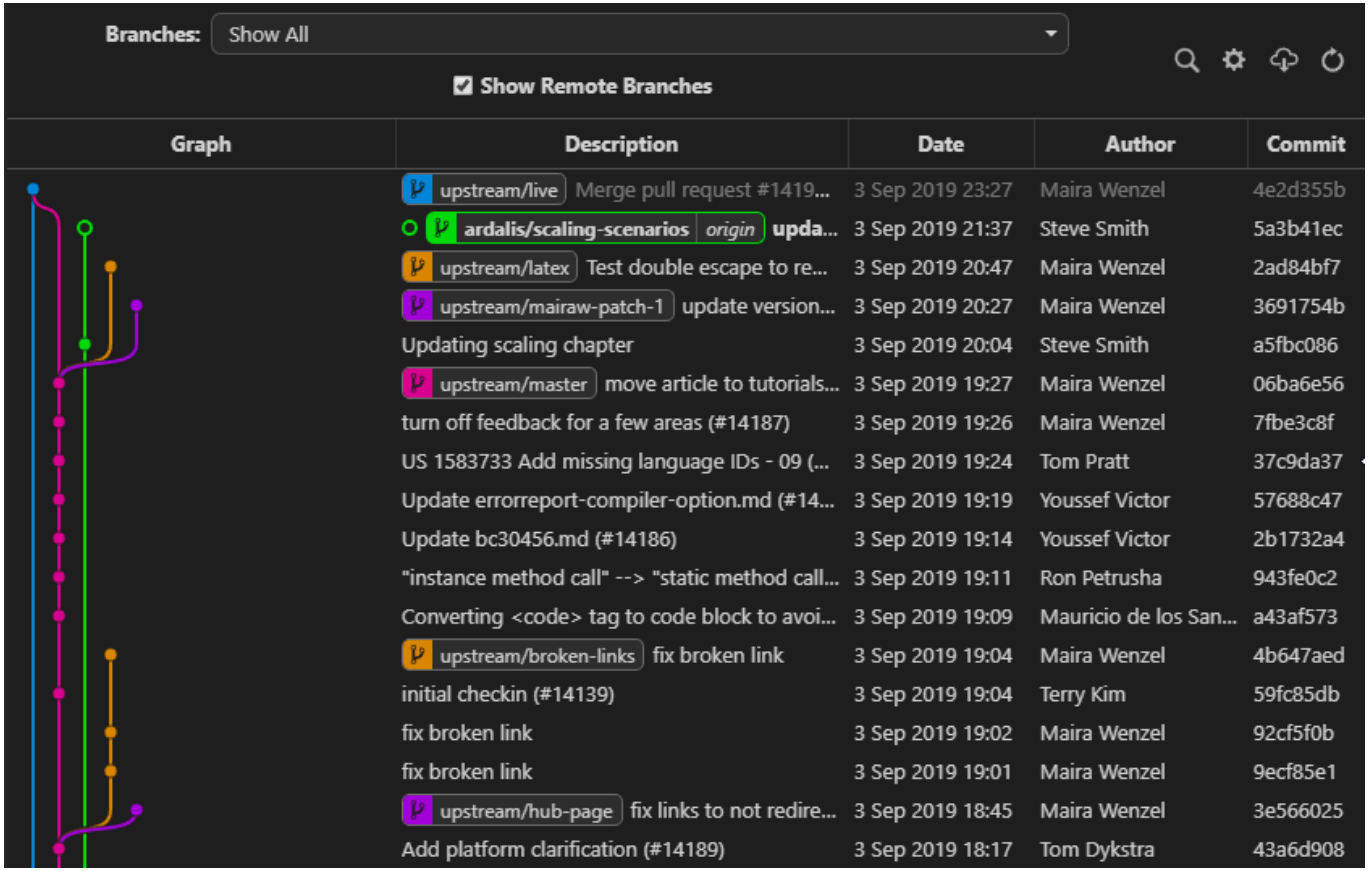
Git takes snapshots



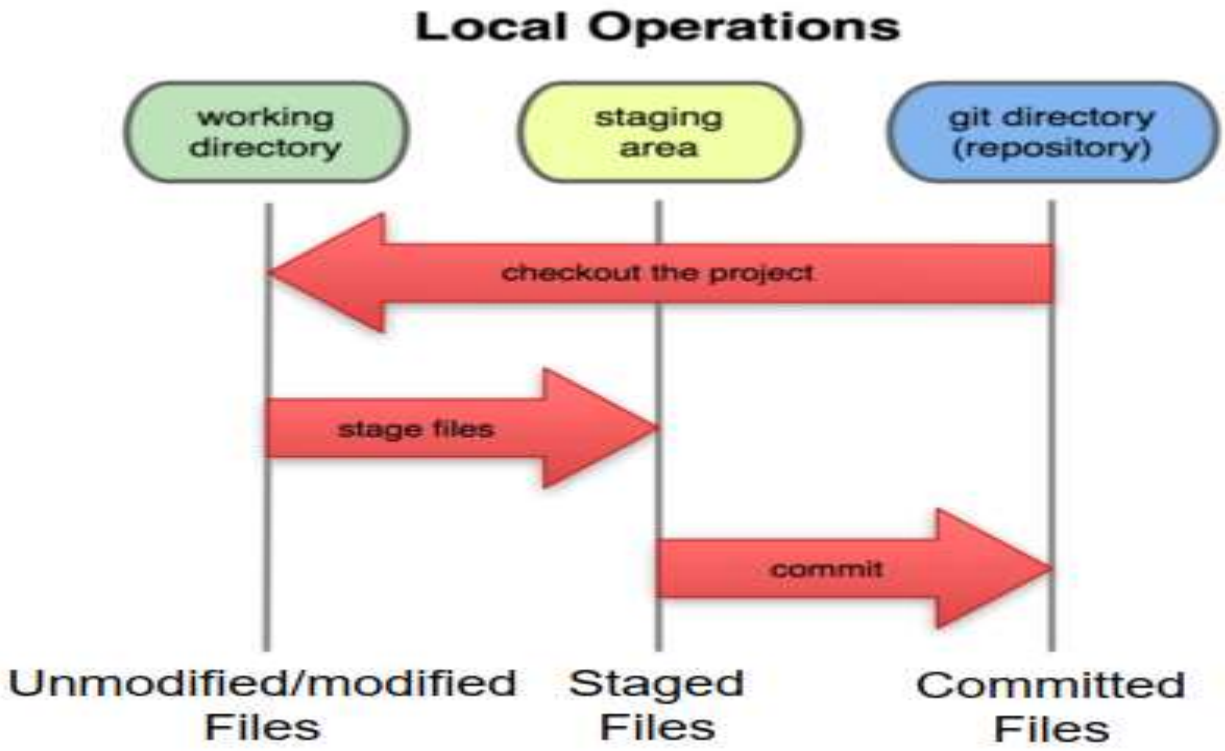
Git takes snapshots



Git takes snapshots



A Local Git project has three areas



Git commands

command	description
`git clone url [dir]`	copy a git repository so you can add to it
`git add files`	adds file contents to the staging area
`git commit`	records a snapshot of the staging area
`git status`	view the status of your files in the working directory and staging area
`git diff`	shows diff of what is staged and what is modified but unstaged
`git help [command]`	get help info about a particular command
`git pull`	fetch from a remote repo and try to merge into the current branch
`git push`	push your new branches and data to a remote repository
others: `init, reset, branch, checkout, merge, log, tag`	

Create a local copy of a repo

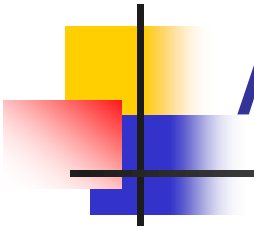
- 2. Two common scenarios: (only do one of these)
  - To **clone an already existing repo** to your current directory:
    `$ git clone <url> [local dir name]`
    This will create a directory named *local dir name*, containing a working copy of the files from the repo, and a **.git** directory (used to hold the staging area and your actual repo)
  - To **create a Git repo** in your current directory:
    `$ git init`
    This will create a **.git** directory in your current directory.

Get ready to use Git!

- 1. Set the name and email for Git to use when you commit:

```
$ git config --global user.name “username”
$ git config --global user.email emailaccount@gmail.com
```

- You can call **git config –list** to verify these are set.
- These will be set globally for all Git projects you work with.
- You can also set variables on a project-only basis by not using the **--global** flag.

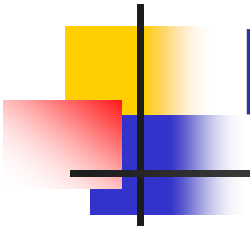


Add files

- Now we have our local repository .
- we can propose changes by *adding* it to the **Index** using the following commands:
- `#For a specific file : $ git add <filename>`
  `#Or to add all files and folders simply use : $ git add .`

Committing files

- Adding the files (proposing the changes) is the first step in the basic git workflow. To actually commit these changes use
- `$ git commit -m “Commit message”`
- Now the file is committed to the **HEAD**, but not in your remote repository yet.

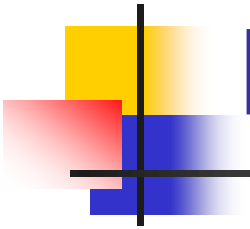


Pulling

- **Pull** from remote repo to get most recent changes (fix conflicts if necessary, add and commit them to your local repo)

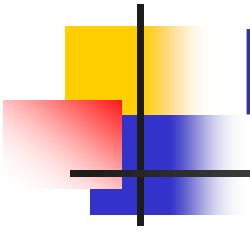
- To fetch the most recent updates from the remote repo into your local repo, and put them into your working directory:

```
$ git pull origin master
```



Pushing

- After **commit** changes are now in the HEAD of your local working copy.
- To send those changes to your remote repository, execute
- `$ git push origin <master>`
- Change #master to whatever branch you want to push your changes to.



Pushing

- Notes: **origin** = an alias for the URL you cloned from **master** = the remote branch you are pulling from/pushing to, (the local branch you are pulling to/pushing from is your current branch)
- If you have not cloned an existing repository and want to connect your repository to a remote server, you need to add it with
- `$ git remote add origin <remote server>`
-

Branching

- To create a new branch named "feature_x" and switch to it using

```
$ git checkout -b feature_x
```

- To switch back to master type:

```
$ git checkout master
```

- To list all branches: (* shows which one you are currently on)

```
$ git branch
```

- After your work is done and you want to delete the branch simply type:

```
$ git branch -d feature_x
```

- A branch will not be available to others unless you push the branch to your remote repository using :

```
$ git push origin <branch>
```

Update & Merge

- In order to update your local repository to the newest commit, type

```
$ git pull
```

- in your working directory to fetch and merge all the remote changes.
- To merge another branch into your active branch (e.g. master), use

```
$ git merge <branch>
```

- before merging changes and fixing the conflicts, you can also preview the differences between the branches by using :

```
$ git diff <source_branch> <target_branch>
```